Given a set of cities $(\{1,\ldots,n\})$ and a symmetric matrix $C=(c_{ij})$, $c_{ij}\geq 0$ that specifies for every pair $(i,j)\in [n]\times [n]$ the cost for travelling from city i to city j. Find a permutation π of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

is minimized.

Theorem 2

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

Hamiltonian Cycle: For a given undirected graph G=(V,E) decide whether ther exists a simple cycle that contains all nodes in G.

Traveling Salesman

Given a set of cities $(\{1,\ldots,n\})$ and a symmetric matrix $C=(c_{ij}),\,c_{ij}\geq 0$ that specifies for every pair $(i,j)\in [n]\times [n]$ the cost for travelling from city i to city j. Find a permutation π of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

is minimized.

Theorem 2

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

Hamiltonian Cycle:

For a given undirected graph G = (V, E) decide whether there exists a simple cycle that contains all nodes in G.

Traveling Salesman

Given a set of cities $(\{1, ..., n\})$ and a symmetric matrix $C = (c_{ij}), c_{ij} \ge 0$ that specifies for every pair $(i, j) \in [n] \times [n]$ the cost for travelling from city i to city j. Find a permutation π of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

is minimized.

15 TSP 332/575

Theorem 2

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

Hamiltonian Cycle:

For a given undirected graph G = (V, E) decide whether there exists a simple cycle that contains all nodes in G.

- Given an instance to HAMPATH we create an instance for TSP.

Traveling Salesman

Given a set of cities $(\{1,\ldots,n\})$ and a symmetric matrix $C = (c_{ij}), c_{ij} \ge 0$ that specifies for every pair $(i, j) \in [n] \times [n]$ the cost for travelling from city i to city j. Find a permutation π of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

15 TSP

is minimized.

Theorem 2

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

Hamiltonian Cycle:

For a given undirected graph G = (V, E) decide whether there exists a simple cycle that contains all nodes in G.

- Given an instance to HAMPATH we create an instance for TSP.
- ▶ If $(i, j) \notin E$ then set c_{ij} to $n2^n$ otw. set c_{ij} to 1. This instance has polynomial size.
- ▶ There exists a Hamiltonian Path iff there exists a tour with cost n. Otw. any tour has cost strictly larger than $n2^n$.
- An $\mathcal{O}(2^n)$ -approximation algorithm could decide btw. these

Traveling Salesman

Given a set of cities $(\{1,\ldots,n\})$ and a symmetric matrix $C=(c_{ij}),\,c_{ij}\geq 0$ that specifies for every pair $(i,j)\in [n]\times [n]$ the cost for travelling from city i to city j. Find a permutation π of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

15 TSP

is minimized.

Theorem 2

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

Hamiltonian Cycle:

For a given undirected graph G = (V, E) decide whether there exists a simple cycle that contains all nodes in G.

- Given an instance to HAMPATH we create an instance for TSP.
- ▶ If $(i, j) \notin E$ then set c_{ij} to $n2^n$ otw. set c_{ij} to 1. This instance has polynomial size.
- There exists a Hamiltonian Path iff there exists a tour with cost n. Otw. any tour has cost strictly larger than $n2^n$.

Traveling Salesman

Given a set of cities $(\{1,\ldots,n\})$ and a symmetric matrix $C = (c_{ij}), c_{ij} \ge 0$ that specifies for every pair $(i, j) \in [n] \times [n]$ the cost for travelling from city i to city j. Find a permutation π of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

15 TSP

is minimized.

Theorem 2

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

Hamiltonian Cycle:

For a given undirected graph G = (V, E) decide whether there exists a simple cycle that contains all nodes in G.

- Given an instance to HAMPATH we create an instance for
- TSP. ▶ If $(i, j) \notin E$ then set $c_{i,i}$ to $n2^n$ otw. set $c_{i,i}$ to 1. This
- instance has polynomial size. There exists a Hamiltonian Path iff there exists a tour with
- cost n. Otw. any tour has cost strictly larger than $n2^n$.

15 TSP

\triangleright An $\mathcal{O}(2^n)$ -approximation algorithm could decide btw. these

Traveling Salesman

Given a set of cities $(\{1,\ldots,n\})$ and a symmetric matrix $C = (c_{ij}), c_{ij} \ge 0$ that specifies for every pair $(i, j) \in [n] \times [n]$ the cost for travelling from city i to city j. Find a permutation π of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

is minimized.

Metric Traveling Salesman

Theorem 2

In the metric version we assume for every triple $i, j, k \in \{1, ..., n\}$

$$c_{ij} \le c_{ij} + c_{jk} .$$

Hamiltonian Cycle:

Traveling Salesman

For a given undirected graph G = (V, E) decide whether there exists a simple cycle that contains all nodes in G.

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

- Given an instance to HAMPATH we create an instance for TSP. ▶ If $(i, j) \notin E$ then set $c_{i,i}$ to $n2^n$ otw. set $c_{i,i}$ to 1. This
- instance has polynomial size.
 - ▶ There exists a Hamiltonian Path iff there exists a tour with
- cost n. Otw. any tour has cost strictly larger than $n2^n$. \blacktriangleright An $\mathcal{O}(2^n)$ -approximation algorithm could decide btw. these

Metric Traveling Salesman

Theorem 2

There does not exist an $O(2^n)$ -approximation algorithm for TSP.

TSP.

In the metric version we assume for every triple $i, j, k \in \{1, ..., n\}$

 $c_{ij} \leq c_{ij} + c_{jk}$.

Hamiltonian Cycle:

Traveling Salesman

For a given undirected graph G = (V, E) decide whether there

It is convenient to view the input as a complete undirected graph

G = (V, E), where c_{ij} for an edge (i, j) defines the distance

exists a simple cycle that contains all nodes in G.

Given an instance to HAMPATH we create an instance for

▶ If $(i, j) \notin E$ then set $c_{i,i}$ to $n2^n$ otw. set $c_{i,i}$ to 1. This

instance has polynomial size.

▶ There exists a Hamiltonian Path iff there exists a tour with

15 TSP

332

cost n. Otw. any tour has cost strictly larger than $n2^n$. \blacktriangleright An $\mathcal{O}(2^n)$ -approximation algorithm could decide btw. these

cases. Hence, cannot exist unless P = NP.

between nodes i and j.

Lemma 3The cost $OPT_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $OPT_{MST}(G)$ of a minimum spanning tree in G.

EADS II

Metric Traveling Salesman

In the metric version we assume for every triple $i,j,k\in\{1,\ldots,n\}$ $c_{ij}\leq c_{ij}+c_{jk}\ .$

It is convenient to view the input as a complete undirected graph G=(V,E), where c_{ij} for an edge (i,j) defines the distance between nodes i and j.

15 TSP

Lemma 3

The cost $OPT_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $OPT_{MST}(G)$ of a minimum spanning tree in G.

Proof:

- ► Take the optimum TSP-tour.
- ► Dolote one edge
- This pieces are entire that of each at most OPT (C)

Metric Traveling Salesman

In the metric version we assume for every triple $i,j,k\in\{1,\ldots,n\}$ $c_{ij}\leq c_{ij}+c_{jk}\ .$

It is convenient to view the input as a complete undirected graph G=(V,E), where c_{ij} for an edge (i,j) defines the distance between nodes i and j.

Lemma 3

The cost $OPT_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $OPT_{MST}(G)$ of a minimum spanning tree in G.

Proof:

- ► Take the optimum TSP-tour.
- Delete one edge.
- This rivers are a first at an extension (C)

Metric Traveling Salesman

In the metric version we assume for every triple $i,j,k\in\{1,\dots,n\}$ $c_{ij}\leq c_{ij}+c_{jk}\ .$

It is convenient to view the input as a complete undirected graph G=(V,E), where c_{ij} for an edge (i,j) defines the distance between nodes i and j.

Lemma 3

The cost $OPT_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $OPT_{MST}(G)$ of a minimum spanning tree in G.

Proof:

- ► Take the optimum TSP-tour.
- Delete one edge. ▶ This gives a spanning tree of cost at most $OPT_{TSP}(G)$.

Metric Traveling Salesman

In the metric version we assume for every triple $i, j, k \in \{1, \ldots, n\}$ $c_{ij} \leq c_{ij} + c_{jk}$.

It is convenient to view the input as a complete undirected graph G = (V, E), where $c_{i,i}$ for an edge (i, j) defines the distance between nodes i and j.

- ► Start with a tour on a subset *S* containing a single node.
- ▶ Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ▶ Reneat until all nodes have been processed

TSP: Lower Bound I

Lemma 3

The cost $\mathrm{OPT}_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $\mathrm{OPT}_{MST}(G)$ of a minimum spanning tree in G.

Proof:

- ► Take the optimum TSP-tour.
- ► Delete one edge.
- ▶ This gives a spanning tree of cost at most $OPT_{TSP}(G)$.

15 TSP

- Start with a tour on a subset S containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ▶ Reneat until all nodes have been processed

TSP: Lower Bound I

Lemma 3

The cost $\mathrm{OPT}_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $\mathrm{OPT}_{MST}(G)$ of a minimum spanning tree in G.

Proof:

- ► Take the optimum TSP-tour.
- ► Delete one edge.
- ▶ This gives a spanning tree of cost at most $OPT_{TSP}(G)$.

- Start with a tour on a subset S containing a single node.
- ▶ Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- Repeat until all nodes have been processed.

TSP: Lower Bound I

Lemma 3

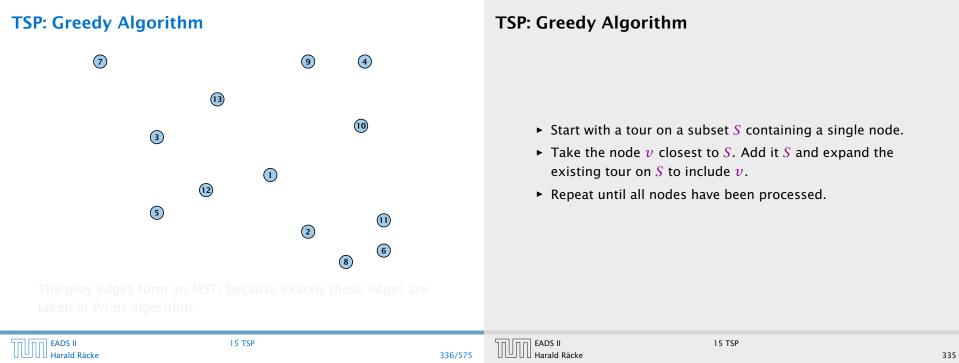
The cost $OPT_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $OPT_{MST}(G)$ of a minimum spanning tree in G.

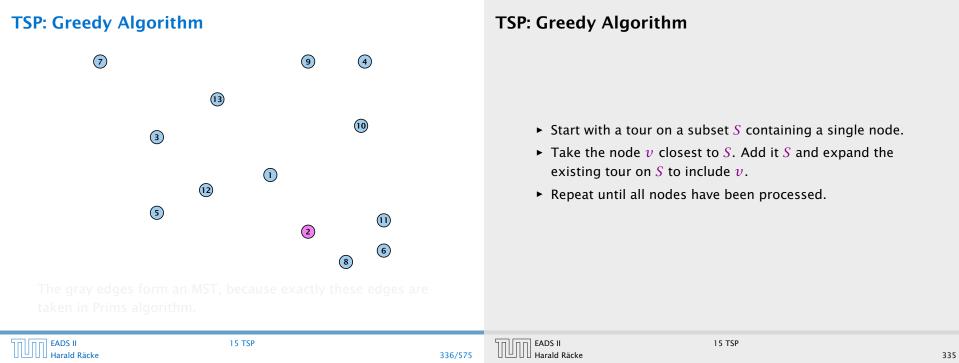
Proof:

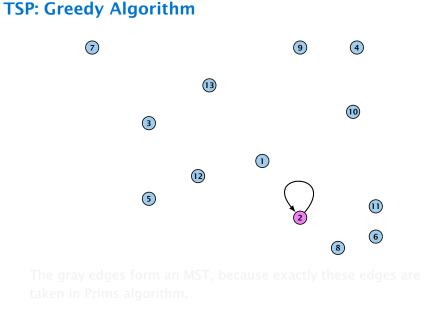
- ► Take the optimum TSP-tour.
- ► Delete one edge.
- ▶ This gives a spanning tree of cost at most $OPT_{TSP}(G)$.

335/575

334





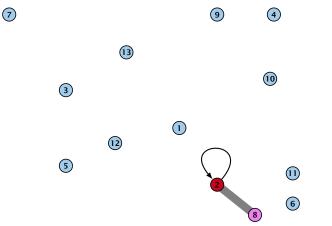


ightharpoonup Start with a tour on a subset $\mathcal S$ containing a single node.

15 TSP

- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

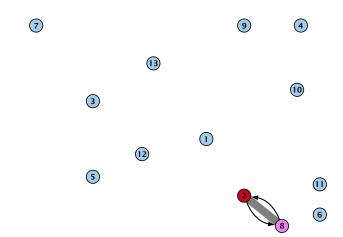
EADS II
Harald Räcke



The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

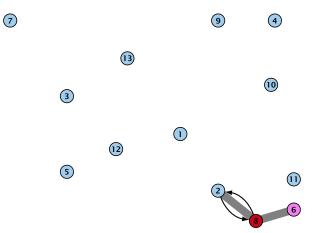
- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.



The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.



The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

TSP: Greedy Algorithm 7 4 13 10 (3) 1 (12) (5)

The gray edges form an MST, because exactly these edges a

TSP: Greedy Algorithm

- \blacktriangleright Start with a tour on a subset S containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

- 7

4

10

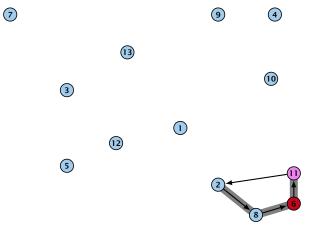
13

(3)

- 1 (12)
- (5)

TSP: Greedy Algorithm

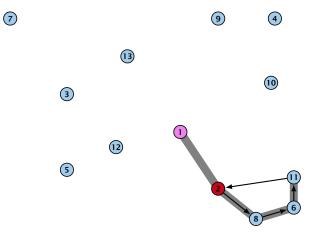
- ▶ Start with a tour on a subset *S* containing a single node.
- \blacktriangleright Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.



The gray edges form an MST, because exactly these edges are

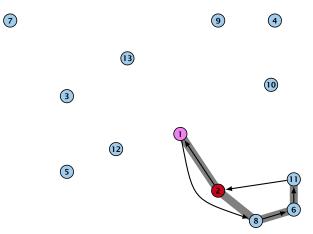
TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ▶ Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.



TSP: Greedy Algorithm

- ▶ Start with a tour on a subset *S* containing a single node.
- \blacktriangleright Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

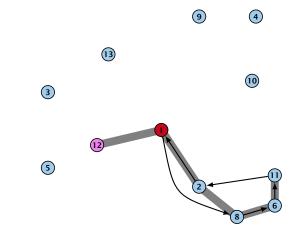


The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

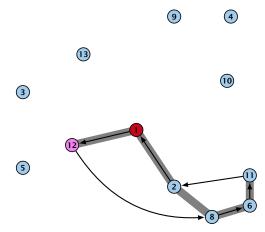


The gray edges form an MST, because exactly these edges are taken in Prims algorithm

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

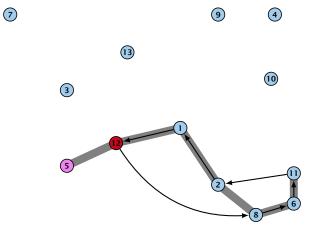
7



The gray edges form an MST, because exactly these edges are taken in Prims algorithm

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

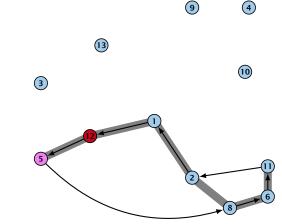


The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

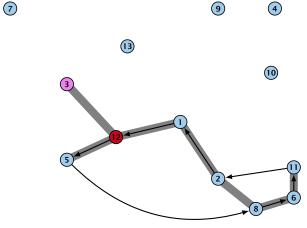
7



The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

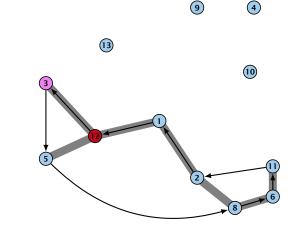


The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

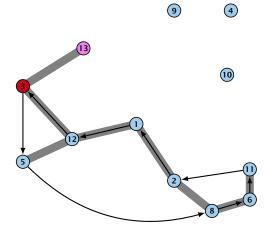


The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

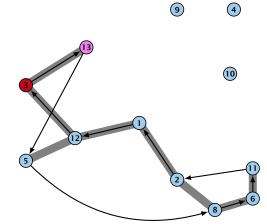


The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

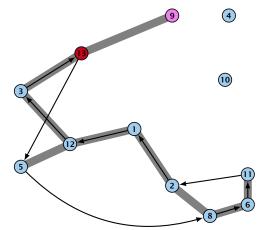


The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

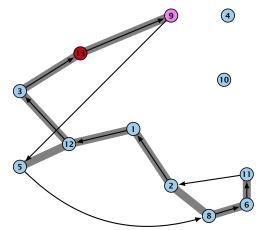


The gray edges form an MST, because exactly these edges are taken in Prims algorithm

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

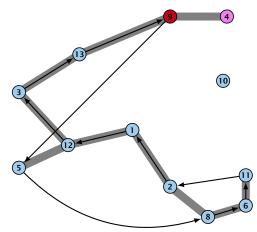


The gray edges form an MST, because exactly these edges are taken in Prims algorithm

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7



The gray edges form an MST, because exactly these edges are taken in Prims algorithm

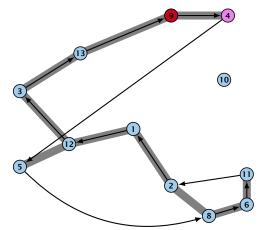
TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

EADS II

335

7

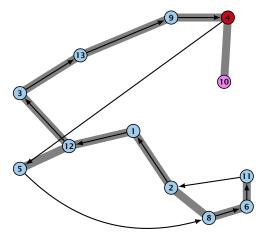


The gray edges form an MST, because exactly these edges are taken in Prims algorithm

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

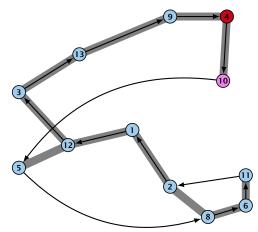


The gray edges form an MST, because exactly these edges are taken in Prims algorithm

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

7

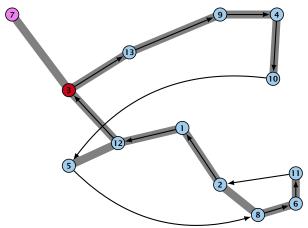


The gray edges form an MST, because exactly these edges are taken in Prims algorithm

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

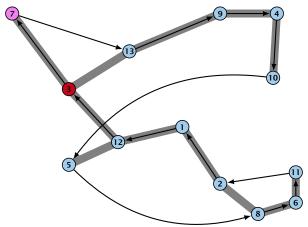
335



The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.



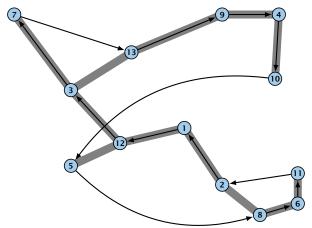
The gray edges form an MST, because exactly these edges are

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ► Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

336/575

EADS II

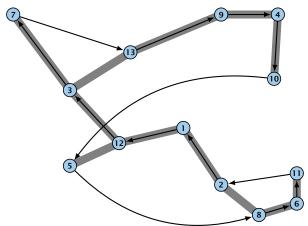


TSP: Greedy Algorithm

- ▶ Start with a tour on a subset *S* containing a single node.
- \blacktriangleright Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

336/575

335



The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

TSP: Greedy Algorithm

- ► Start with a tour on a subset *S* containing a single node.
- ▶ Take the node v closest to S. Add it S and expand the existing tour on S to include v.
- ► Repeat until all nodes have been processed.

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the i-th iteration, and let v denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$

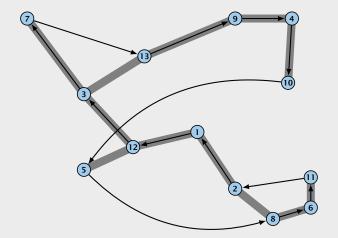
Let r_i denote the successor of s_i in the tour before inserting v_i

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i) and (v_i, r_i) .

This increases the cost by

$$c_{s_i,v_i} + c_{v_i,r_i} - c_{s_i,r_i} \le 2c_{s_i,v_i}$$

TSP: Greedy Algorithm



The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the *i*-th iteration, and let v_i denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

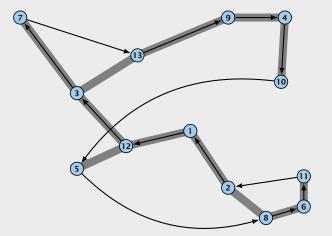
Let r_i denote the successor of s_i in the tour before inserting v_i

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i) and (v_i, r_i) .

This increases the cost by

$$c_{S_i,v_i} + c_{v_i,r_i} - c_{S_i,r_i} \le 2c_{S_i,v_i}$$

TSP: Greedy Algorithm



The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the *i*-th iteration, and let v_i denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

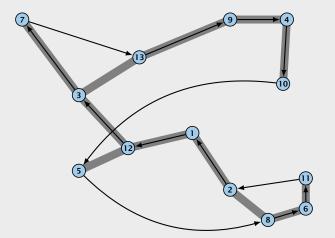
Let r_i denote the successor of s_i in the tour before inserting v_i

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i) and (v_i, r_i) .

This increases the cost by

$$c_{S_i,v_i} + c_{v_i,r_i} - c_{S_i,r_i} \le 2c_{S_i,v_i}$$

TSP: Greedy Algorithm



The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

15 TSP

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the i-th iteration, and let v_i denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

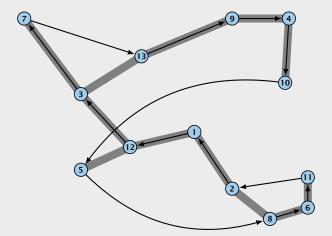
Let r_i denote the successor of s_i in the tour before inserting v_i .

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i) and (v_i, r_i) .

This increases the cost by

$$c_{\mathcal{S}_i,v_i} + c_{v_i,r_i} - c_{\mathcal{S}_i,r_i} \le 2c_{\mathcal{S}_i,v_i}$$

TSP: Greedy Algorithm



The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

15 TSP

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the i-th iteration, and let v_i denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

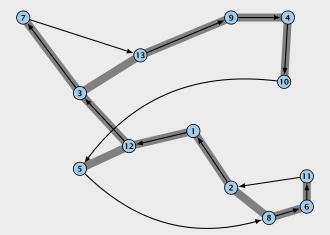
Let r_i denote the successor of s_i in the tour before inserting v_i .

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i) and (v_i, r_i) .

This increases the cost by

 $c_{s_i,v_i} + c_{v_i,r_i} - c_{s_i,r_i} \le 2c_{s_i,v_i}$

TSP: Greedy Algorithm



The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the i-th iteration, and let v_i denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

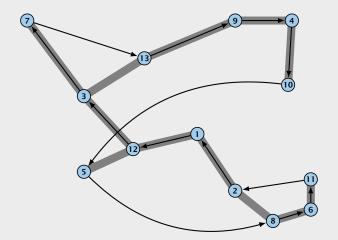
Let r_i denote the successor of s_i in the tour before inserting v_i .

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i) and (v_i, r_i) .

This increases the cost by

$$c_{s_i,v_i} + c_{v_i,r_i} - c_{s_i,r_i} \le 2c_{s_i,v_i}$$

TSP: Greedy Algorithm



The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

The edges (s_i, v_i) considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

$$\sum_{i} c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$$

TSP: Greedy Algorithm

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the *i*-th iteration, and let v_i denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

Let r_i denote the successor of s_i in the tour before inserting v_i .

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i)

This increases the cost by

$$c_{S_i,v_i} + c_{v_i,r_i} - c_{S_i,r_i} \le 2c_{S_i,v_i}$$

and (v_i, r_i) .

The edges (s_i, v_i) considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

Hence,

$$\sum_{i} c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$$

which with the previous lower bound gives a 2-approximation.

TSP: Greedy Algorithm

Lemma 4

The Greedy algorithm is a 2-approximation algorithm.

Let S_i be the set at the start of the *i*-th iteration, and let v_i denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

Let r_i denote the successor of s_i in the tour before inserting v_i .

15 TSP

We replace the edge (s_i, r_i) in the tour by the two edges (s_i, v_i) and (v_i, r_i) .

This increases the cost by

$$c_{s_i,v_i} + c_{v_i,r_i} - c_{s_i,r_i} \le 2c_{s_i,v_i}$$

and a TSP-tour of cost at most
$$\sum c'(e)$$

TSP: Greedy Algorithm

The edges (s_i, v_i) considered during the Greedy algorithm are

Hence,

$$\sum_{i} c_{s_i, v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$$

exactly the edges considered during PRIMs MST algorithm.

which with the previous lower bound gives a 2-approximation.

15 TSP

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

$$\sum_{e \in E'} c'(e)$$

15 TSP

TSP: Greedy Algorithm

The edges (s_i, v_i) considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

 $\sum c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$

which with the previous lower bound gives a 2-approximation.

15 TSP

338

339/575

EADS II

Hence,

EADS II

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

TSP: Greedy Algorithm

Hence,

338

 $\sum c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$

which with the previous lower bound gives a 2-approximation.

The edges (s_i, v_i) considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in F'} c'(e)$$

- Find an Fuler tour of G'.

15 TSP

TSP: Greedy Algorithm

The edges (s_i, v_i) considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

Hence,

$$\sum_{i} c_{s_i, v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$$

which with the previous lower bound gives a 2-approximation.

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.

TSP: Greedy Algorithm

The edges (s_i, v_i) considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

Hence,

$$\sum_{i} c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$$

which with the previous lower bound gives a 2-approximation.

15 TSP

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.

TSP: Greedy Algorithm

The edges (s_i, v_i) considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

Hence,

$$\sum_{s} c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$$

15 TSP

which with the previous lower bound gives a 2-approximation.

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- Find an Fuler tour of G'. Fix a permutation of the cities (i.e., a TSP-tour) by traversing
- the Euler tour and only note the first occurrence of a city. ▶ The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

TSP: Greedy Algorithm

The edges (s_i, v_i) considered during the Greedy algorithm are

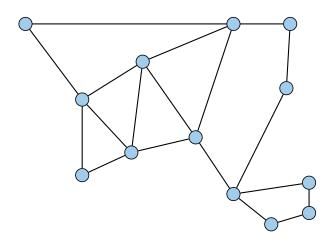
Hence,

 $\sum c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$

exactly the edges considered during PRIMs MST algorithm.

which with the previous lower bound gives a 2-approximation.

This technique is known as short cutting the Euler tour.



TSP: A different approach

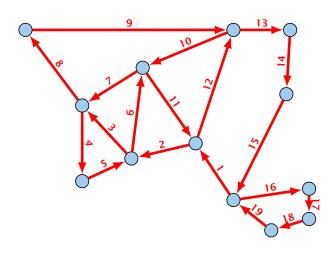
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

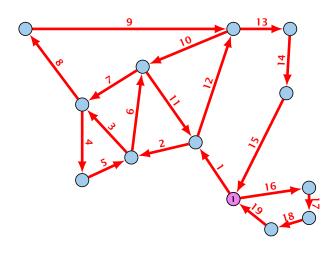
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

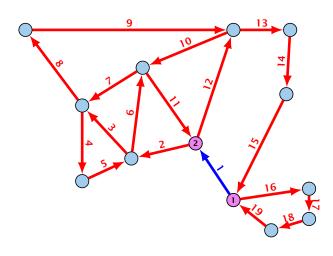
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

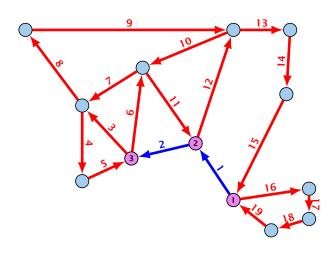
$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ▶ The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.

340/575

339



TSP: A different approach

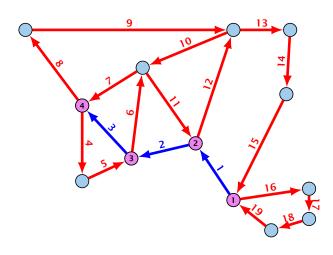
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

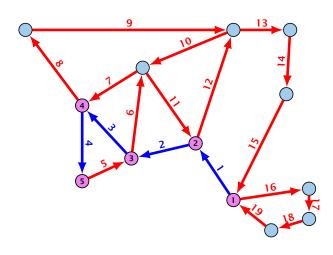
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

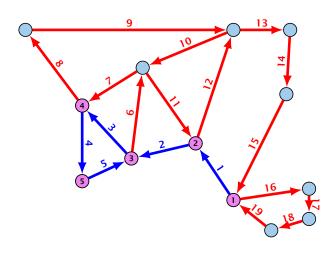
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

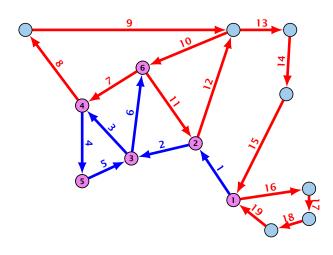
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

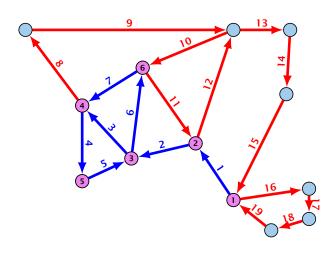
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

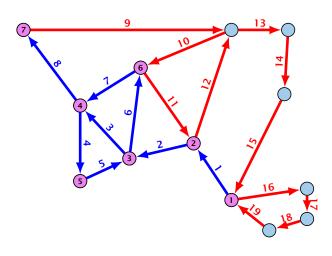
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

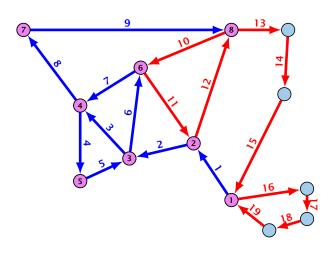
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

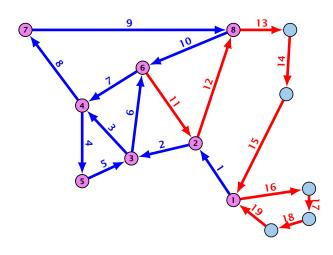
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



TSP: A different approach

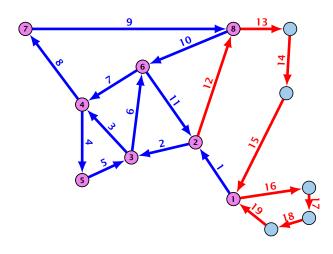
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

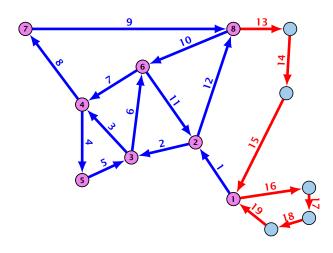
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

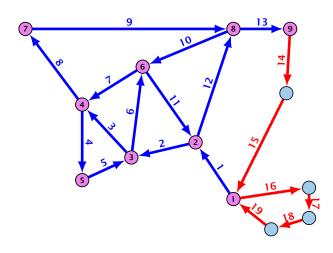
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

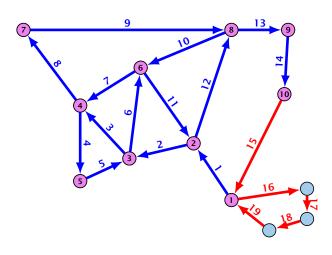
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

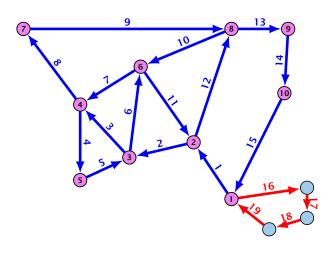
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

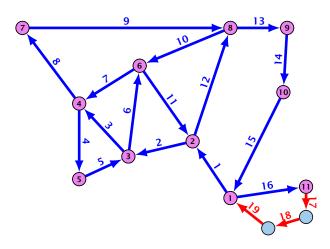
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



TSP: A different approach

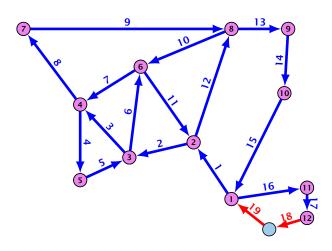
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

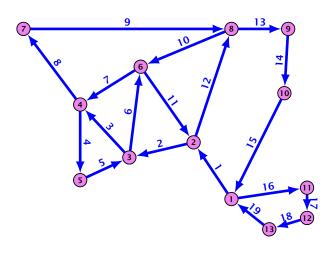
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

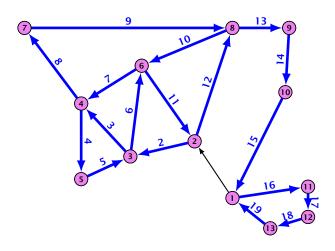
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

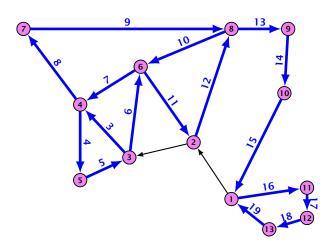
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

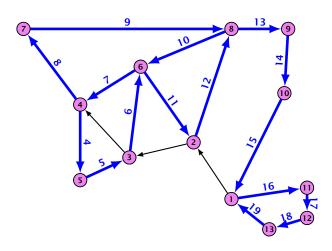
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ▶ Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ▶ The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

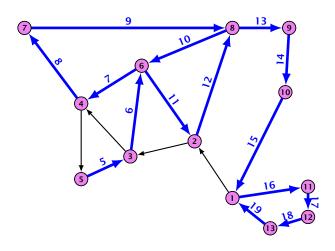
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

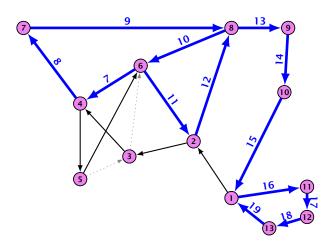
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

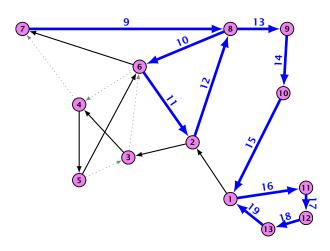
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



TSP: A different approach

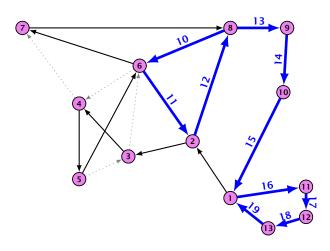
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

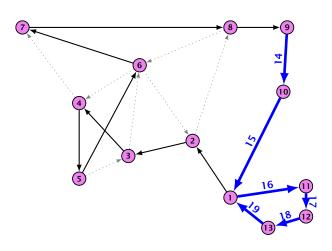
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Euler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

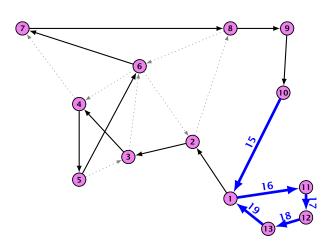
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

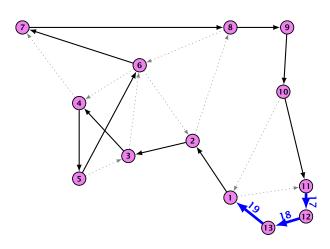
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

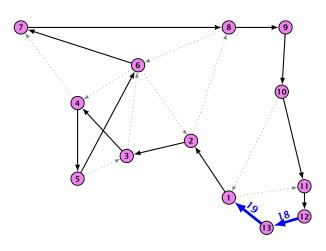
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

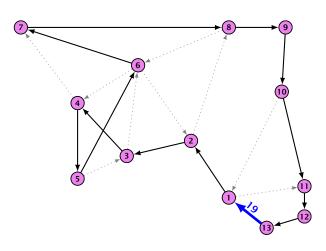
Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

15 TSP

This technique is known as short cutting the Euler tour.



TSP: A different approach

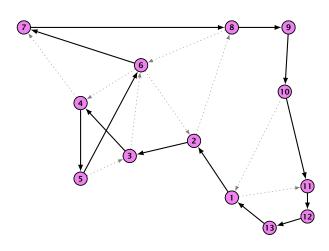
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

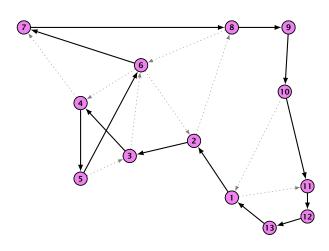
Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.



15 TSP

TSP: A different approach

Suppose that we are given an Eulerian graph G' = (V, E', c') of G = (V, E, c) such that for any edge $(i, j) \in E'$ $c'(i, j) \ge c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

- ► Find an Fuler tour of G'.
- ► Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
- ► The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.

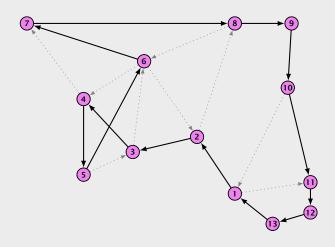
Consider the following graph:

- ▶ Compute an MST of *G*.
- Duplicate all edges.

This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.

Hence, short-cutting gives a tour of cost no more than $2 \cdot OPT_{MST}(G)$ which means we have a 2-approximation

TSP: A different approach





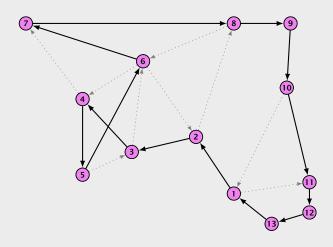
Consider the following graph:

- ► Compute an MST of *G*.
- Duplicate all edges.

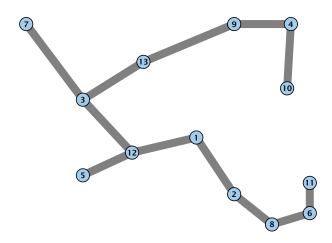
This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.

Hence, short-cutting gives a tour of cost no more than $2 \cdot OPT_{MST}(G)$ which means we have a 2-approximation.

TSP: A different approach





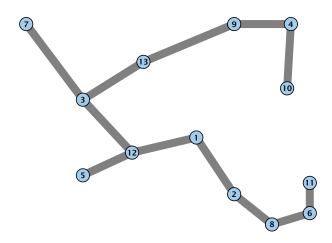


TSP: A different approach

Consider the following graph:

- ► Compute an MST of *G*.
- ► Duplicate all edges.

This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.

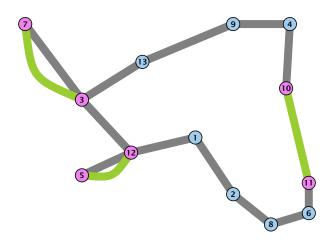


TSP: A different approach

Consider the following graph:

- ► Compute an MST of *G*.
- ► Duplicate all edges.

This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.

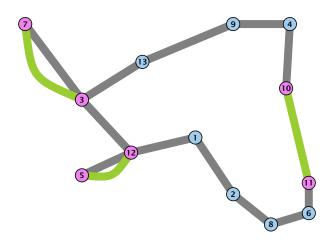


TSP: A different approach

Consider the following graph:

- ► Compute an MST of *G*.
- ► Duplicate all edges.

This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.

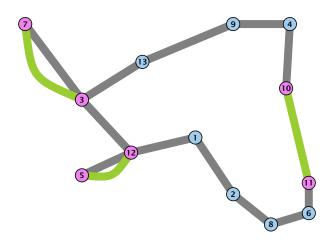


TSP: A different approach

Consider the following graph:

- ► Compute an MST of *G*.
- ► Duplicate all edges.

This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.

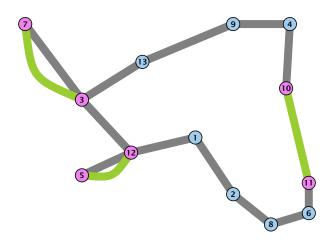


TSP: A different approach

Consider the following graph:

- ► Compute an MST of *G*.
- ► Duplicate all edges.

This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.



TSP: A different approach

Consider the following graph:

- ► Compute an MST of *G*.
- ► Duplicate all edges.

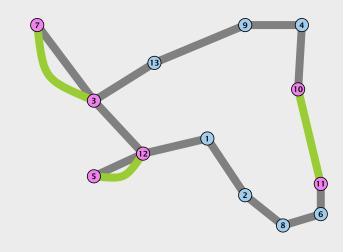
This graph is Eulerian, and the total cost of all edges is at most $2 \cdot OPT_{MST}(G)$.

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).

TSP: Can we do better?



15 TSP

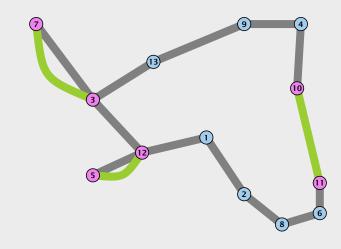


Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Fulerian

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them)

TSP: Can we do better?



□ EADS II

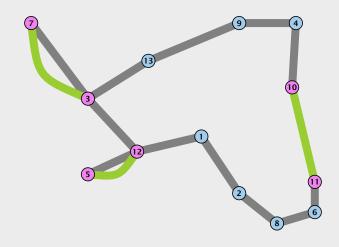
342

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them)

TSP: Can we do better?



15 TSP

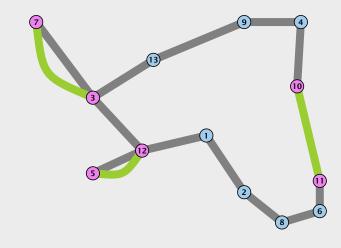
EADS II

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).

TSP: Can we do better?



343/575

342

An optimal tour on the odd-degree vertices has cost at most $OPT_{TSP}(G)$.

However, the edges of this tour give rise to two disjoint matchings. One of these matchings must have weight less than $\mathrm{OPT}_{\mathrm{TSP}}(G)/2$.

Adding this matching to the MST gives an Eulerian graph with edge weight at most

15 TSP

This is the best that is kno

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).



EADS II

Harald Räcke

An optimal tour on the odd-degree vertices has cost at most $OPT_{TSP}(G)$.

TSP: Can we do better?

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).

15 TSP

15 TSP

An optimal tour on the odd-degree vertices has cost at most $OPT_{TSP}(G)$.

However, the edges of this tour give rise to two disjoint matchings. One of these matchings must have weight less than $OPT_{TSP}(G)/2$.

$$OPT_{MST}(G) + OPT_{TSP}(G)/2 \le \frac{3}{2}OPT_{TSP}(G)$$
,

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).

An optimal tour on the odd-degree vertices has cost at most $OPT_{TSP}(G)$.

However, the edges of this tour give rise to two disjoint matchings. One of these matchings must have weight less than $\mathrm{OPT}_{\mathrm{TSP}}(G)/2$.

Adding this matching to the MST gives an Eulerian graph with edge weight at most

$$OPT_{MST}(G) + OPT_{TSP}(G)/2 \le \frac{3}{2}OPT_{TSP}(G)$$
,

Short cutting gives a $\frac{3}{2}$ -approximation for metric TSP

This is the best that is known

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).

An optimal tour on the odd-degree vertices has cost at most $OPT_{TSP}(G)$.

However, the edges of this tour give rise to two disjoint matchings. One of these matchings must have weight less than $\mathrm{OPT}_{\mathrm{TSP}}(G)/2$.

Adding this matching to the MST gives an Eulerian graph with edge weight at most

$$OPT_{MST}(G) + OPT_{TSP}(G)/2 \le \frac{3}{2}OPT_{TSP}(G)$$
,

Short cutting gives a $\frac{3}{2}$ -approximation for metric TSP.

This is the host that is known

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).

An optimal tour on the odd-degree vertices has cost at most $OPT_{TSP}(G)$.

However, the edges of this tour give rise to two disjoint matchings. One of these matchings must have weight less than $OPT_{TSP}(G)/2$.

Adding this matching to the MST gives an Eulerian graph with edge weight at most

$$OPT_{MST}(G) + OPT_{TSP}(G)/2 \le \frac{3}{2}OPT_{TSP}(G)$$
,

Short cutting gives a $\frac{3}{2}$ -approximation for metric TSP.

TSP: Can we do better?

Duplicating all edges in the MST seems to be rather wasteful.

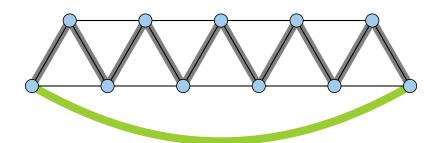
We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number

of them).

15 TSP

Christofides. Tight Example



- optimal tour: n edges.
- ▶ MST: n-1 edges.
- weight of matching (n+1)/2-1
- ► MST+matching $\approx 3/2 \cdot n$

TSP: Can we do better?

An optimal tour on the odd-degree vertices has cost at most $\mathrm{OPT}_{\mathrm{TSP}}(G)$.

However, the edges of this tour give rise to two disjoint matchings. One of these matchings must have weight less than $\mathrm{OPT}_{\mathrm{TSP}}(G)/2$.

Adding this matching to the MST gives an Eulerian graph with edge weight at most

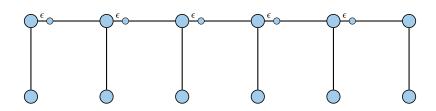
$$OPT_{MST}(G) + OPT_{TSP}(G)/2 \le \frac{3}{2}OPT_{TSP}(G)$$
,

15 TSP

Short cutting gives a $\frac{3}{2}$ -approximation for metric TSP.

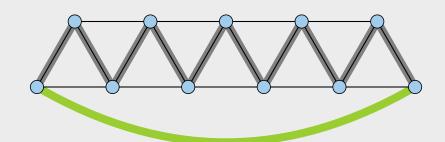
This is the best that is known.

Tree shortcutting. Tight Example



• edges have Euclidean distance.

Christofides. Tight Example

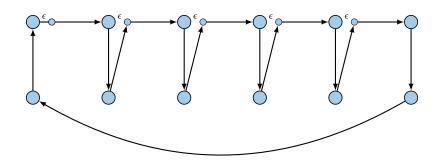


- ▶ optimal tour: n edges.
- ► MST: n-1 edges.
- weight of matching (n+1)/2-1
- ► MST+matching $\approx 3/2 \cdot n$

346/575

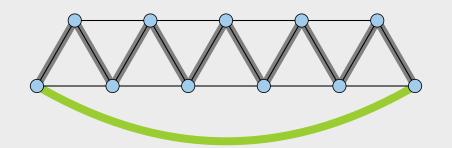
345

Tree shortcutting. Tight Example



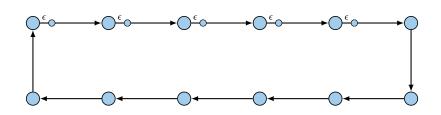
• edges have Euclidean distance.

Christofides. Tight Example



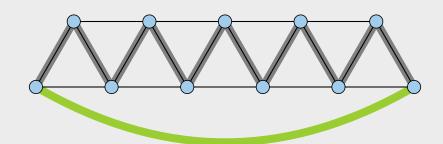
- ► optimal tour: *n* edges.
- ► MST: n-1 edges.
- weight of matching (n+1)/2-1
- ► MST+matching $\approx 3/2 \cdot n$

Tree shortcutting. Tight Example



edges have Euclidean distance.

Christofides. Tight Example



- ► optimal tour: *n* edges.
- ► MST: n-1 edges.
- weight of matching (n+1)/2-1
- ► MST+matching $\approx 3/2 \cdot n$